Analytical Performance Comparison of BNP Scheduling Algorithms

Gagandeep Kaur[1], Er. Navneet Singh[2] and Parneet Kaur[3]

[1]

Received: 9 December 2011 Accepted: 1 January 2012 Published: 15 January 2012

Abstract

Parallel computing is related to the application of many computers running in parallel to solve computationally intensive problems. One of the biggest issues in parallel computing is efficient task scheduling. In this paper, we survey the algorithms that allocate a parallel program represented by an edge-directed acyclic graph (DAG) to a set of homogenous processors with the objective of minimizing the completion time. We examine several such classes of algorithms and then compare the performance of a class of scheduling algorithms known as the bounded number of processors (BNP) scheduling algorithms. Comparison is based on various scheduling parameters such as makespan, speed up, processor utilization and scheduled length ratio. The main focus is given on measuring the impact of increasing the number of tasks and processors on the performance of these four BNP scheduling algorithms.

***Index terms*—** Parallel computing, Scheduling, DAG, Homogeneous processors.

1 Introduction

arallel computing is a technique of executing multiple tasks simultaneously on multiple processors. The main goal of parallel computing is to increase the speed of computation. Efficient task scheduling & mapping is one of the biggest issue in homogeneous parallel computing environment [1]. The objective of Scheduling is to manage the execution of tasks in such a way that certain optimality criterion is met. Most scheduling algorithms are based on listscheduling technique [4] [6][2] [11]. There are two phases in List-scheduling technique: task prioritizing phase, where the priority is computed and assigned to each node in DAG, and a processor selection phase, where each task in is assigned to a processor in order of the priority of nodes that minimizes a suitable cost function. List scheduling algorithms are classified as static list scheduling if the processor selection phase starts after completion of the task prioritizing phase and dynamic list scheduling algorithm if the two phases are interleaved. A parallel program can be represented by a node-and edge-weighted directed acyclic graph (DAG) [2] [3]. The Directed Acyclic Graph is a generic model of a parallel program consisting of a set of processes. The nodes represent the application process and the edges represent the data dependencies among these processes.

This paper surveys various scheduling algorithms that schedule an edge-weighted directed acyclic graph (DAG), which is also called a task graph, to a set of homogeneous processors. We examine four classes of algorithms: Bounded Number of Processors (BNP) scheduling algorithms, Unlimited Number of Clusters (UNC) scheduling algorithms, and Arbitrary Processor Network (APN) & Task Duplication Based (TDB) scheduling algorithms. Performance comparisons are made for the BNP algorithms. We provide qualitative analyses by measuring the performance of these four BNP scheduling algorithms under useful scheduling parameters: makespan, speed up, processor utilization, and scheduled length ratio.

The rest of this paper is organized as follows. In the next section, we describe the generic DAG model and discuss its variations & techniques. A classification of scheduling algorithms is presented in Section 3.The four BNP scheduling algorithms are discussed in Section 4.The performance results and comparisons are presented in Section 5, Section 6 concludes the paper. Section 7 suggest about future scope of research.

2 II.

3 Task scheduling problem & model used

This section presents the application model used for task scheduling. The number of processors could be limited or unlimited. The homogeneous computing environment model is used for the surveyed algorithms. We first introduce the directed acyclic graph (DAG) model of a parallel program. This is followed by a discussion about some basic techniques used in most scheduling algorithms & homogeneous computing environment.

4 a) The DAG Model

The Directed Acyclic Graph [2][3] is a generic model of a parallel program consisting of a set of processes among which there are dependencies. The DAG model that we use within this analysis is presented below in Fig. 1 A task without any parent is called an entry task and a task without any child is called an exit task. A node cannot start execution before it gathers all of the messages from its parent nodes. The communication cost between two tasks assigned to the same processor is assumed to be zero. If node ni is scheduled to some processor, then ST(ni) and FT(ni)denote the start-time and finish-time of ni, respectively.After all the nodes have been scheduled, the schedule length is defined as maxi{FT(ni)}across all processors. The node-and edge-weights are usually obtained by estimation. Some variations in the generic DAG model are:-Accurate model [2][3]-In an accurate model, the weight of a node includes the computation time, the time to receive messages before the computation, and the time to send messages after the computation. The weight of an edge is a function of the distance between the source and the destination nodes. It also depends on network topology and contention which can be difficult to model. When two nodes are assigned to a single processor, the edge weight becomes zero. Approximate model 1 [2][3] -Here the edge weight is approximated by a constant. A completely connected network without contention fits this model. Approximate model 2 [2][3]-In this model, the message receiving time and sending time are ignored in addition to approximating the edge weight by a constant.

An accurate model is useless when the weights of nodes and edges are not accurate. As the node and edge weights are obtained by estimation, which is hardly accurate, the approximate models are used. The approximate models can be used for medium to large granularity, since the larger the process grain-size, the less the communication, and consequently the network is not heavily loaded.

Preemptive scheduling: The preemptive scheduling is prioritized. The highest priority process should always be the process that is currently utilized.

Non-Preemptive scheduling: When a process enters the state of running, the state of that process is not deleted from the scheduler until it finishes its service time.

The homogeneous computing environment model is a set P of p identical processors connected in a fully connected graph [4]. It is also assumed that:

Any processor can execute the task and communicate with other processors at the same time.

Once a processor has started task execution, it continues without interruption, and on completing the execution it sends immediately the output data to all children tasks in parallel.

5 b) Basic Techniques in DAG Scheduling

Most scheduling algorithms are based on list scheduling. The basic idea of list scheduling is to assign priorities to the nodes of DAG, then place the nodes in a list called ready list according to the priority levels and then lastly map the nodes onto the processors in the order of priority. A higher priority node will be examined first for scheduling before a node with a lower priority. In case any two or more nodes have the same priority, then the ties are needed to be break using some useful method. There are various ways to determine the priorities of nodes such as HLF (Highest level First), LP (Longest Path), LPT (Longest Processing Time) and CP (Critical Path).Frequently used attributes for assigning priority are [2][4] [5]:t-level: t-level(Top Level) of the node ni in DAG is the length of the longest path from entry node to ni (excluding ni) i.e. the sum of all the nodes computational costs and edges weights along the path.

b-level: The b-level (Bottom Level)of a node ni is the length of the longest path from node ni to an exit node . The b-level is computed recursively by traversing the DAG upward starting from the exit node.

Static level: Some scheduling algorithms do not consider the edge weights in computing the b-level known as static b-level. or static level.

ALAP time: The ALAP (As-Late-As-Possible) start time of a node is measure of how far the node's start time can be delayed without increasing the schedule length. It is also known as latest start time (LST).

CP (Critical Path):It is the length of the longest path from entry node to the exit node A DAG can have more than one CP. b-level of a node is bounded by the length of a critical path.

EST (Earliest Starting Time): Procedure for computing the t-levels can also be used to compute the EST of nodes. The other name for EST is ASAP (As-Soon-As-Possible) start-time.

A DAG -G = (V, E, w, c) -that represents the application to be scheduled

6 Bnp scheduling algorithms

In this section, we discuss four basic BNP scheduling algorithms: HLFET, ISH, MCP, and ETF. All these algorithms are for a limited number of homogeneous processors. The major characteristics of these algorithms are summarized in Table 1 [6]. In table, p denotes the number of processors given.

7 Performance Results and Comparison

In this section, we present the performance results and comparisons of the 4 BNP scheduling algorithms discussed above. The comparisons are based upon the following four comparison metrics [2][4]: 1. Makespan: Makespan is defined as the completion time of the algorithm. It is calculated by measuring the finishing time of the exit task by the algorithm. 2. Speed Up: The Speed Up value is computed by dividing the sequential execution time by the parallel execution time.

1) Calculate the static b-level of each node.

2) Make a ready list in a descending order of static b-level. Initially, the ready list contains only the entry nodes. Ties are broken randomly. Repeat 3) Schedule the first node in the ready list to a processor that allows the earliest execution, using the non-insertion approach. 4) Update the ready list by inserting the nodes that are now ready. Until all nodes are scheduled.

8 1) Calculate the static b-level of each node.

2) Make a ready list in a descending order of static b-level. Initially, the ready list contains only the entry nodes. Ties are broken randomly. Repeat 3) Schedule the first node in the ready list to the processor that allows the earliest execution, using the non-insertion algorithm. 4) If scheduling of this node causes an idle time slot, then find as many nodes as possible from the ready list that can be scheduled to the idle time slot but cannot be scheduled earlier on other processors. 5) Update the ready list by inserting the nodes that are now ready. Until all nodes are scheduled 1) Compute the ALAP time of each node.

2) For each node, create a list which consists of the ALAP times of the node itself and all its children in a descending order. 3) Sort these lists in an ascending lexicographical order. Create a node list according to this order. Repeat 4) Schedule the first node in the node list to a processor that allows the earliest execution, using the insertion approach. 5) Remove the node from the node list. Until the node list is empty.

9 1) Compute the static b-level of each node.

2) Initially, the pool of ready nodes includes only the entry nodes. Repeat 3) Calculate the earliest start-time on each processor for each node in the ready pool. Pick the node-processor pair that gives the earliest time using the non-insertion approach. Ties are broken by selecting the node with a higher static b-level. Schedule the node to the corresponding processor. 4) Add the newly ready nodes to the ready node pool. Until all nodes are scheduled. So it can be concluded that for small number of tasks (35) MCP is the best algorithm but, with increasing number of tasks (50 & 65) ISH is one of the efficient algorithm, considering the data gathered using the scenarios and the performance calculated from them.

Future Scope: A lot of work can be done considering more case scenarios:

The number of tasks can be changed to create test case scenarios. Heterogeneous environment can be considered. [1]

1

Figure 1: Fig. 1 :

Analytical Performance Comparison of BNP Scheduling Algorithms

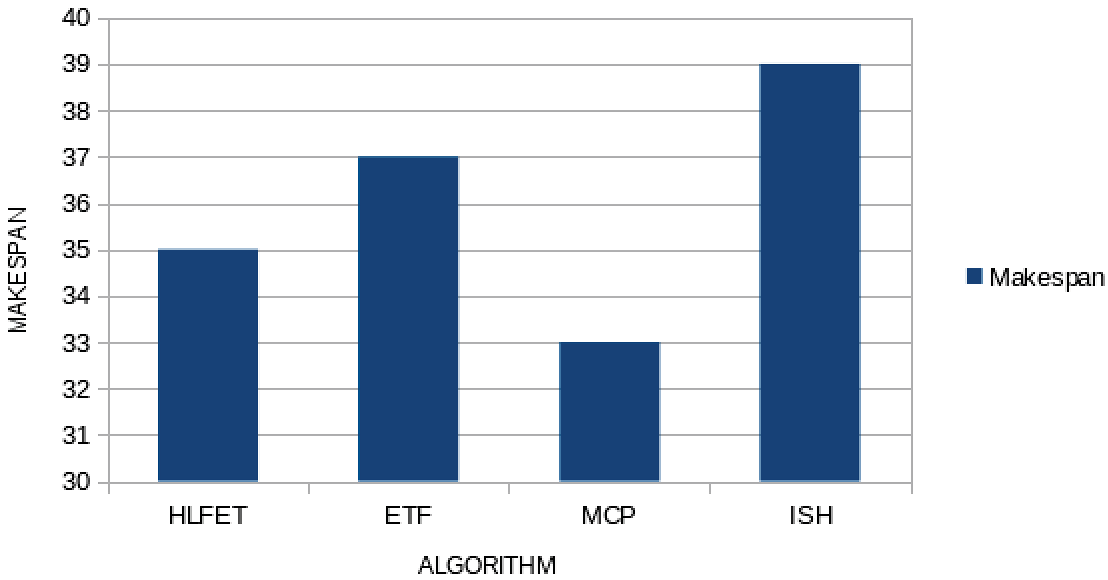
Figure 2: V

2 Analytical Performance Comparison of BNP Scheduling Algorithms

Figure 3: Fig. 2 :

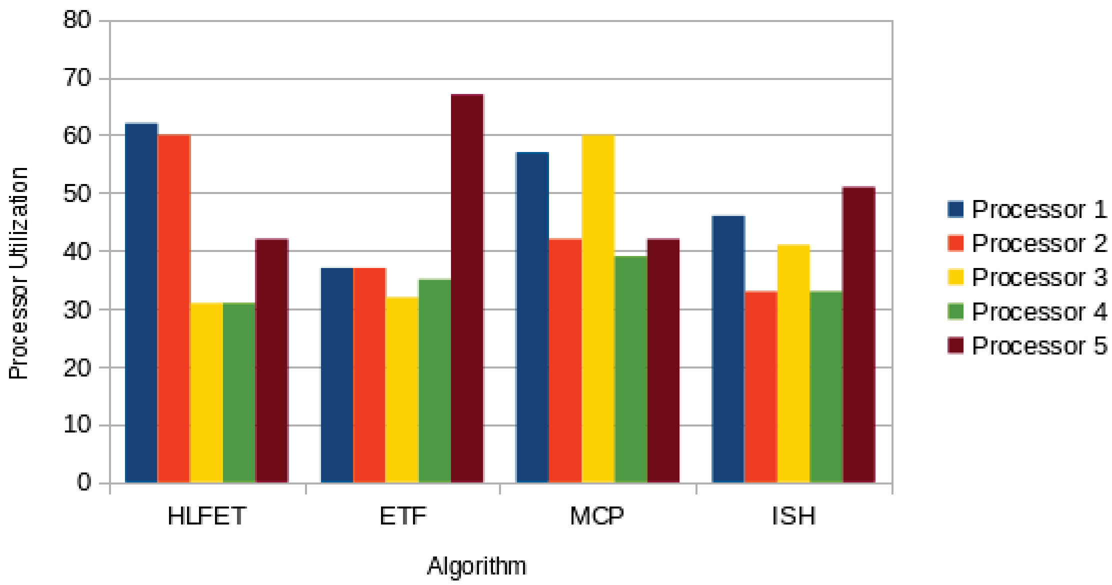
3 Analytical Performance Comparison of BNP Scheduling Algorithms

Figure 4: Fig. 3 :



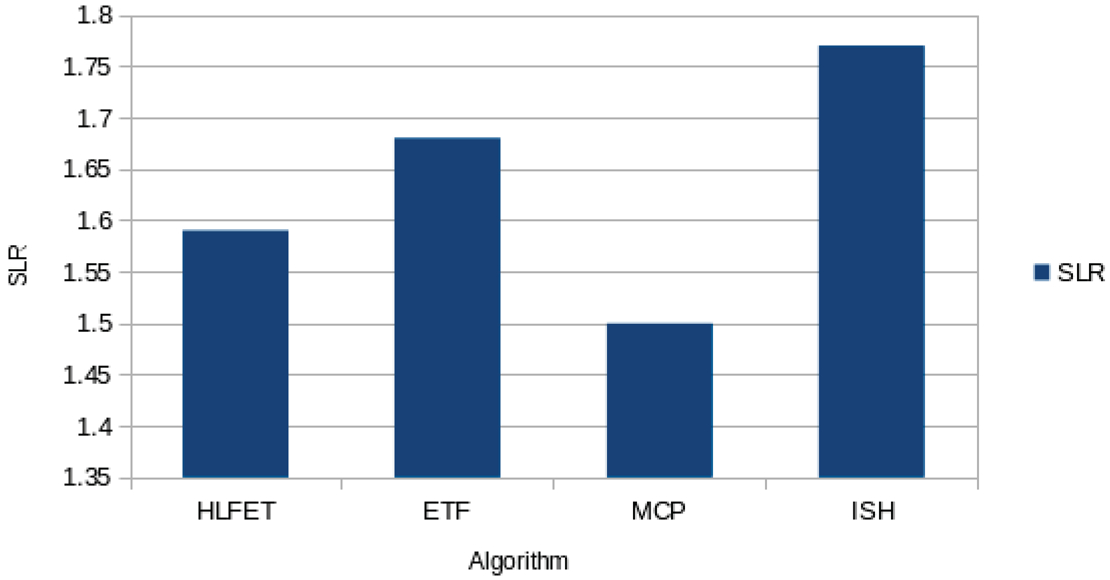
455

Figure 5: Fig. 4 : 5 .Fig. 5 :



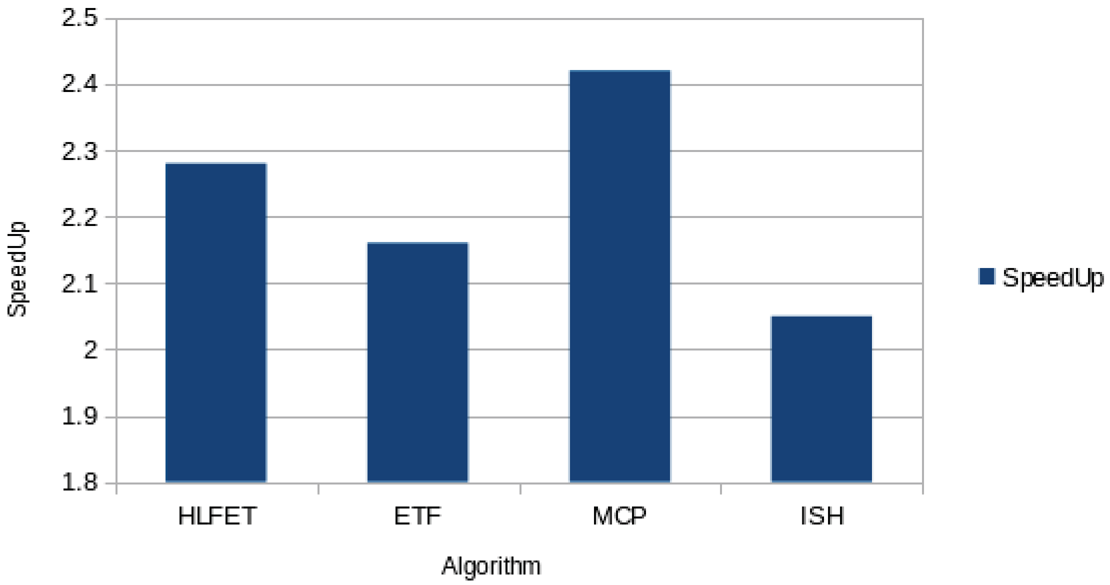
3

Figure 6: 3 .



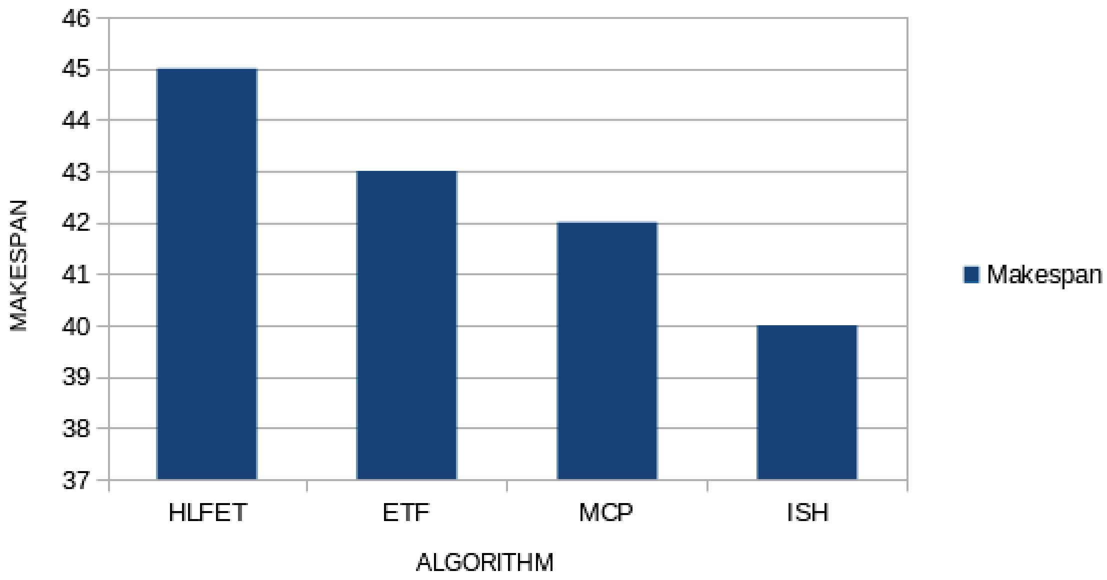
6

Figure 7: Fig. 6 :



89

Figure 8: Fig. 8 :Fig. 9 :

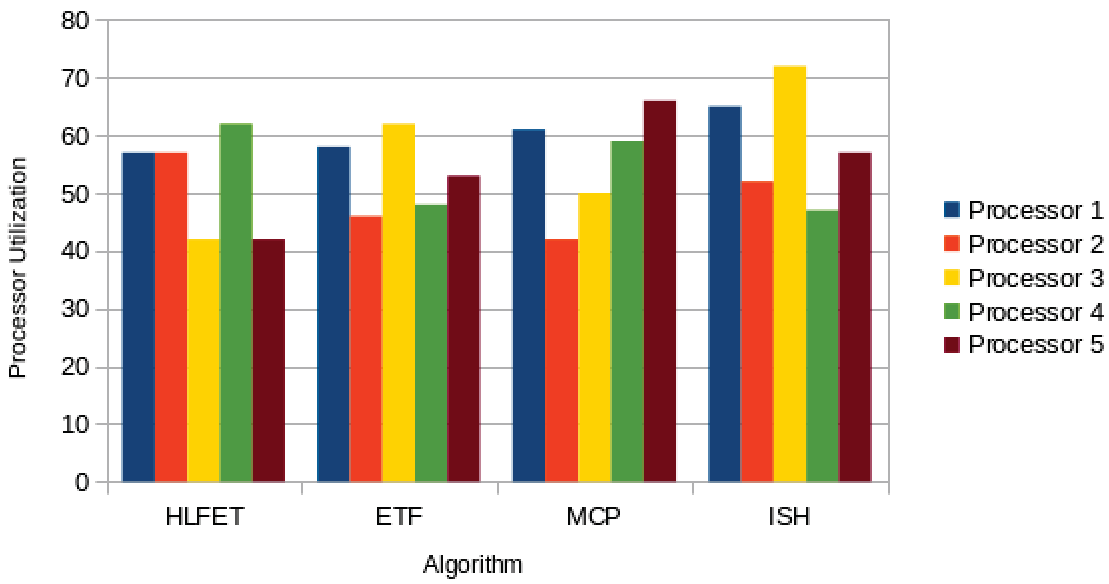


101112

Figure 9: Fig. 10 :Fig. 11 :Fig. 12 :

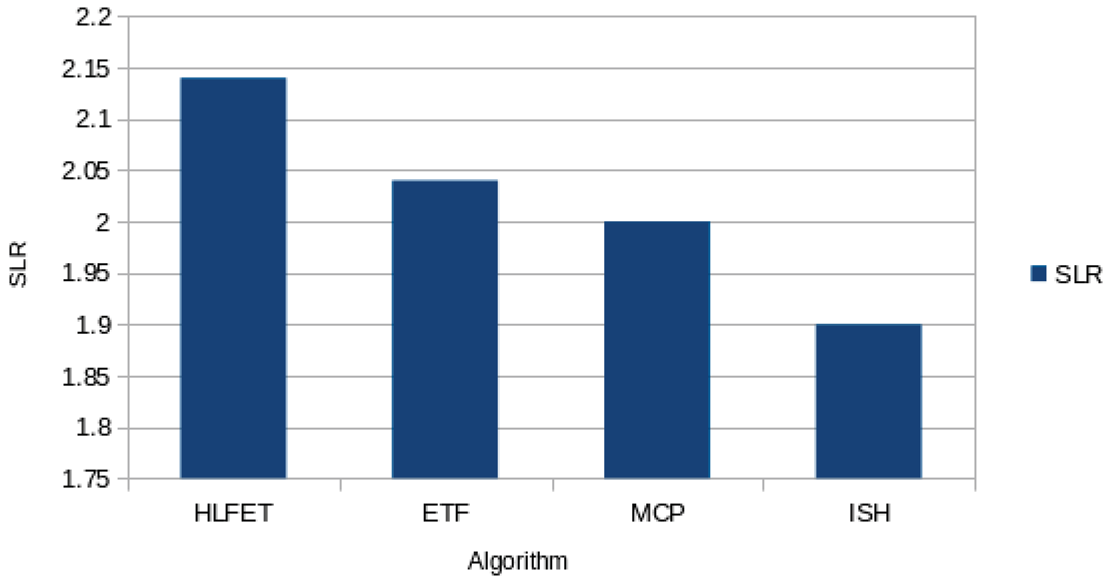
14 ANALYTICAL PERFORMANCE COMPARISON OF BNP SCHEDULING ALGORITHMS

Figure 10: Fig. 14 :



151718

Figure 11: Fig. 15 :Fig. 17 :Fig. 18 :



1920

Figure 12: Fig. 19 :Fig. 20 :

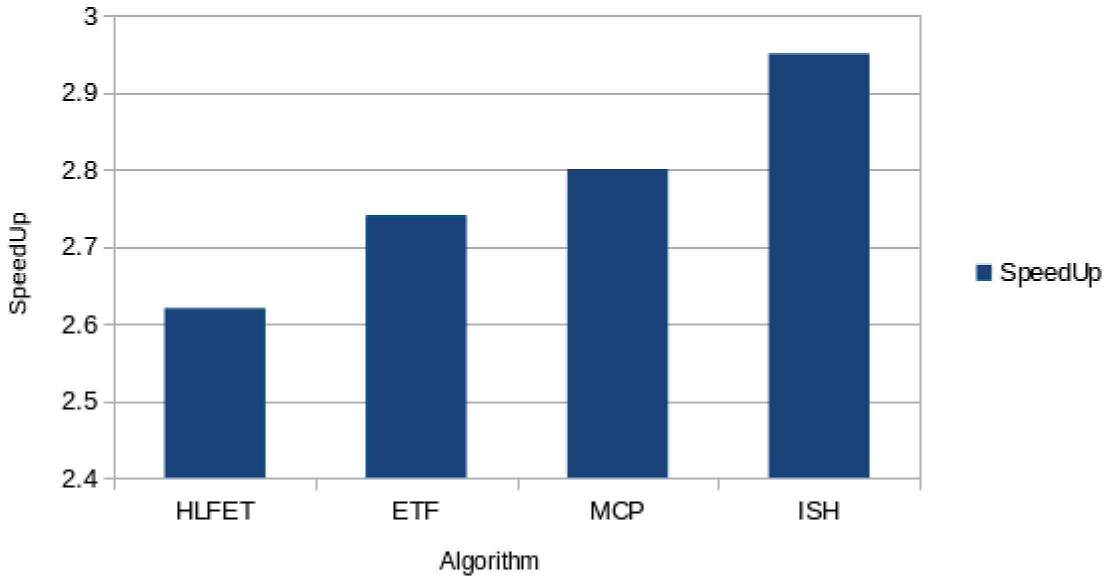


Figure 13: Both

1

Algorithm	Proposed by[year]	Priority List	Type	Greedy
HLFET	Adam et al. [1974]	SL	Static	Yes
ISH	Kruatrachue & Lewis [1987]	SL	Static	Yes
MCP	Wu & Gajski [1990]	ALAP	Static	Yes
ETF	Hwang et al. [1989]	SL	Static	Yes

[Note: a) The HLFET (Highest Level First with Estimated Times) Algorithm[12]: It is one of the simplest scheduling algorithms. The algorithm is briefly described below in Fig.2.]

Figure 14: Table 1 :

350
300
Speed Up 200 HLFET
250
Average 100 MCP ETF
150
50 ISH
0
35 50 65 2012
Number Of Task Nodes

Fig. 21 : Graph representing Speedup of Algorithms Year

When the tasks are 35, The MCP algorithm yields highest Speedup value and ISH gives lowest speedup.

VI. Conclusion and Future Scope

After Comparative analysis following results were derived:

D D D D) A
(

With 35 task nodes, the MCP algorithm gives lowest SLR value with ISH algorithm giving highest SLR value.

With 50 tasks, the ISH shows the lowest SLR value and HLFET gives highest SLR value.

With 65 tasks, the ISH has the lesser SLR values and ETF gives highest value.

d) Average Speedup: Higher the value of Speedup, more efficient is the algorithm. Fig. 21 shows the Speedup of the all 4 algorithms with various nodes cases.

Figure 15:

[Parallelism and Scalability ()] , Parallelism , Programmability Scalability . 1993. NY: Mc Graw Hill.

[Adam et al. (1974)] ‘A Comparison of List Scheduling for Parallel Processing Systems’. T L Adam , K Chandy , J Dickson . *Communications of the ACM* Dec.1974. 17 (12) p. .

[Sharma et al. (2012)] ‘A STUDY OF BNP PARALLEL TASK SCHEDULING ALGORITHMS METRIC’S FOR DISTRIBUTED DATABASE SYSTEM’. Manik Sharma , Dr , Gurdev Singh , Harsimran Kaur . *International Journal of Distributed and Parallel Systems (IJDPS)* January 2012. 3 (1) .

[Hwang] *Advanced Computer Architecture*, K Hwang .

[Lenstra and Kan (1981)] ‘An Introduction to multiprocessor Scheduling Algorithm’. J K Lenstra , A H G Kan . *Questi* March 1981. 5.

[Kaur et al.] ‘Analysis Comparison and Performance Evaluation of BNP Scheduling Algorithm in parallel Processing’. Parneet Kaur , Dheerandra Singh , Gurvinder Singh . *International journal of Knowledge engineering*

[Ahmad et al. ()] *Analysis, Evaluation, and Comparison of Algorithms for Scheduling Task Graphs on Parallel Processors*, Ishfaq Ahmad , Yu-Kwong Kwok , Min-You Wu . 1996. IEEE. p. .

[Kwok and Ahmed ()] ‘Benchmarking the Task Graph Scheduling Algorithms’. Y Kwok , I Ahmed . *Proc. IPPS/SPDP*, (IPPS/SPDP) 1998.

[Droro et al.] G Droro , Larry Feitelson , Uwe Rudolph , Kenneth C Schwiegelshohn , Parkson Sevcik , Wong . *Theory and Practice in Parallel Job Scheduling*,

[Barney] *Introduction to Parallel Computing*, Blaise Barney . Lawrence Livermore National Laboratory

[Ahmad et al.] *Performance Comparison of Algorithms for Static scheduling of DAG to Multi-processors*, Ishfaq Ahmad , Yu-Kwong Kwok , Min-You Wu . `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.42.8979&rep=rep1&type=pdf`

[Kwok and Ahmad (1999)] ‘Static Scheduling Algorithm for Allocating Directed Task Graph to multiprocessors’. Yu-Kwong Kwok , Ishfaq Ahmad . *ACM Computing Surveys* December 1999. 31 (4) .

[Hagras and Janecek ()] ‘Static versus Dynamic List-Scheduling Performance Comparison’. T Hagras , J Janecek . *Acta Polytechnica* 2003. 43 (6) .